```
# -- 1.
# -- a.
# data LinkedList = EmptyList | ListNode Integer LinkedList deriving Show
# ll_contains :: LinkedList -> Integer -> Bool

# ll_contains EmptyList _ = False
# ll_contains (ListNode val rLst) target
#         | val == target = True
#         | otherwise = ll_contains rLst target




# -- b.
# -- data LinkedList = EmptyList | ListNode Integer LinkedList deriving Show
# {-
# Have an integer for indexing and one for the actual value?
# This will allow the traversal to match to the indexing index and then insert at the desired
zero-based index.
# -}

# -- c.
# ll_insert :: LinkedList -> Integer -> Integer -> LinkedList
# -- when inserting a new node, insert the actual node not the entire linked list
# ll_insert EmptyList val _ = ListNode val EmptyList -- Insert at the beginning (for an empty list
or negative index)
# ll_insert (ListNode curr_val rLst) desired_val index
#     | index <= 0 = ListNode desired_val (ListNode curr_val rLst) -- Insert at the beginning (for a
positive index)
#     | otherwise = ListNode curr_val (ll_insert rLst desired_val (index - 1)) -- Recursively insert at
the specified index


# -- 2.
# -- a. Box's value attribute was mutated because the parameter of the box_quintuple attribute
that was passed was the actual box object reference.
# -- Within the box_quintuple function, the box object's value attribute is called and the actual
value within the box object declared outside is
# -- modified because the object reference was passed not a local variable.
# -- Meanwhile the num variable that was declared and then assigned a value wasn't mutated
because the function quintuple was operated on a local variable, not the num variable declared
globally.
# -- A local variable was created when the num was passed but since the function doesn't return
any value or modify the original variable, the 3 will remain the same.
```

-- b.
-- i. After we run the main function, I believe that the program will output the following:
{-
joke6
joke7
joke4
-}

-- ii. With the changes made, the program will print
{-
joke1
joke2
-}

-- 3. Python allows you to pass an object of type Foo or str to len but not an int because an int is technically not an object.
-- Meanwhile strings and other custom objects are actual objects being passed into the len function.The length of an integer
-- is atomic and the len method is only supposed to work on sequeneces or collections. Thus python type checking checks for objects or objects of sequences to find a length.

-- 5.
-- a.
```
def largest_sum(nums, k):
    if k < 0 or k > len(nums):
        raise ValueError
    elif k == 0:
        return 0
    max_sum = None
    for i in range(len(nums)-k+1):
        sum = 0
    for num in nums[i:i+k]:
        sum += num
    if sum > max_sum:
        max_sum = sum
    return max_sum
```
-- b.
{-
```
def largest_sum(nums, k):
```

```
    if k < 0 or k > len(nums):
        raise ValueError
    elif k == 0:
        return 0
    sum = 0
    for num in nums[:k]:
        sum += num
    max_sum = sum
    for i in range(0, len(nums)-k-1):
        sum -= nums[i]
        sum += nums[i+k]
        max_sum = max(sum, max_sum)
    return max_sum
# -}


# 6.
# a
class Event:
    def __init__(self, start_time, end_time):
        if start_time >= end_time:
            raise ValueError
        else:
            self.start_time = start_time
            self.end_time = end_time



# b.
class Calendar:
    def __init__(self):
        self.__events = []

    def get_events(self):
        return self.__events

    def add_events(self, Event):
        if type(Event) != Event:
            raise TypeError
        else:
            self.__events.append(Event)

# c.
class AdventCalendar(Calendar):
```

```
    def __init__(self, year):
        self.year = year

advent_calendar = AdventCalendar(2022)
print(advent_calendar.get_events())

# the reason that the error occurs is because the __events variable is a private variable and
thus cannot be accessed
# by functions outside of the Calendar function even if the events class is inherited.
# 1. To fix the code within the subclass, you would need to establish getters and setters within
the AdventCalendar class
# that are specifically tasked with retrieving the value of events
# 2. Another way to fix the code is to call the parent class initializer so that it can maintain the
same attributes as
# the calendar and then the get_events method can then retrieve the value.
```